学好node开发

Leo Hui

Published
with GitBook

目錄

1. 介紹
2. 搭建开发环境
   i. 官方安装
   ii. 调试方式
3. npm使用
   i. mac全局安装的权限问题
   ii. 常用npm包
   iii. 常见报错及解决方案
4. Meteor
   i. 搭建Meteor开发环境
   ii. 认识Meteor文件结构
5. 附录
   i. node相关库
   ii. MongoDB相关库
   iii. 图书资料
   iv. 常用工具

学好node开发

记录一份node的学习手册。

搭建开发环境

任何语言都需要一个平台，就连运行javascript，你都要开一个浏览器，搭建开发环境使我们学习的第一步。

官方安装

我学习新东西的观念就是去官方的网站找向导。所以在初级学习的阶段，我还是推荐大家根据官方的说明去下载安装。毕竟是最简单最直接的方式。如果你需要多版本的切换，当然选择`nvm`了，后面我也会提到。

不过我相信，大部分人使用一个版本应该就够了，特别是初期！

官网

nodejs.org,首页中间部分就有一个大大的install。点击下载，然后就和其他所有应用软件一样运行安装即可。

安装完成，终端输入`node --version`就可以看到当前的版本。如果提示你未找到命令，检查下你的系统路径，将node所在路径添加进去即可。

调试方式

参考资料

- 翻译 - NodeJS错误处理最佳实践

npm使用

mac全局安装的权限问题

正常用户安装之后，使用`npm install`就是在当前项目中创建`node_modules`。

`npm install -g`的形式，就是创建全局的`node_modules`，一般是供用户全局使用的命令。

出现需要**root**权限的情况

有时候`npm install -g`的时候，会报错:

```
npm ERR! Error: EACCES, open '/Users/{$USER}/.npm/debug/0.7.4/package/package.json'
npm ERR!  { [Error: EACCES, open '/Users/{$USER}/.npm/debug/0.7.4/package/package.json']
npm ERR!   errno: 3,
npm ERR!   code: 'EACCES',
npm ERR!   path: '/Users/{$USER}/.npm/debug/0.7.4/package/package.json',
npm ERR!   parent: 'sohunews-node' }
npm ERR!
npm ERR! Please try running this command again as root/Administrator.

npm ERR! System Darwin 13.2.0
npm ERR! command "node" "/usr/local/bin/npm" "install"
npm ERR! cwd /Users/{$USER}/project/sohu/sohunews-node
npm ERR! node -v v0.10.29
npm ERR! npm -v 1.4.14
npm ERR! path /Users/{$USER}/.npm/debug/0.7.4/package/package.json
npm ERR! code EACCES
npm ERR! errno 3
npm ERR! stack Error: EACCES, open '/Users/{$USER}/.npm/debug/0.7.4/package/package.json'
npm ERR!
npm ERR! Additional logging details can be found in:
npm ERR!     /Users/{$USER}/project/sohu/sohunews-node/npm-debug.log
npm ERR! not ok code 0
```

提示你某个目录当前用户无法在其中无法创建项目。

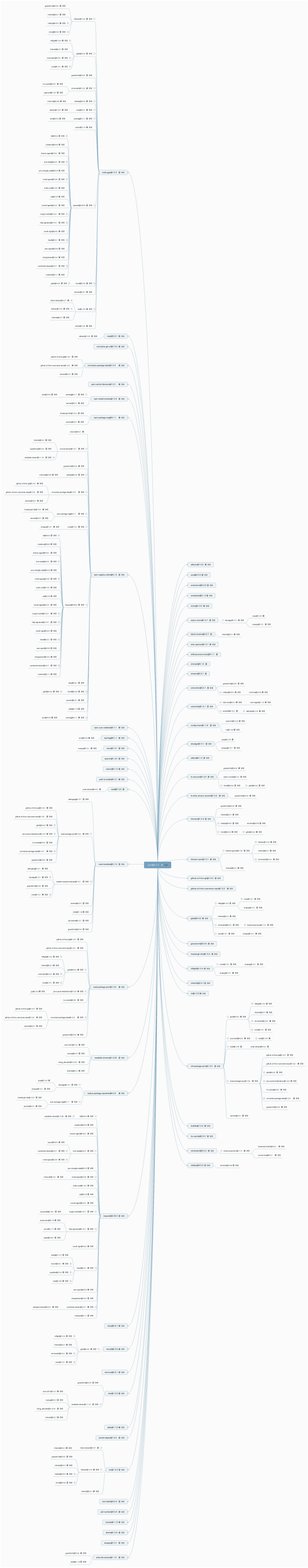
原因是执行.npm目录是在**root**用户下创建的，而我当前执行npm install的用户是非root用户。

解决方法：把根目录下的.npm目录权限更改为当前用户的权限就OK了。命令行输入`sudo chown -R $USER ~/.npm`

现在再执行npm install应该就可以正常安装项目依赖了。

如果再不行，清空一下tmp目录：`rmdir ~/tmp`.

常用npm包



常见报错及解决方案

node-gyp

node-gyp: Node.js native addon build tool.npm中有很多的库依赖了此编译工具，比如 `gulp-sass`，`browser-sync`...而此工具又需要 `Visual C++` 的 `VCBuild.exe` 编译，所以官方给出的依赖环境(on Windows)如下:

```
- Python (v2.7.3 recommended, v3.x.x is not supported)
    - Make sure that you have a PYTHON environment variable, and it is set to drive:\path
- Windows XP/Vista/7:
    - Microsoft Visual Studio C++ 2013 (Express version works well)
        - If the install fails, try uninstalling any C++ 2010 x64&x86 Redistributable tha
        - If you get errors that the 64-bit compilers are not installed you may also need
- Windows 7/8:
    - Microsoft Visual Studio C++ 2013 for Windows Desktop (Express version works well)
- All Windows Versions
    - For 64-bit builds of node and native modules you will also need the Windows 7 64-bi
    - You may need to run one of the following commands if your build complains about Win
        call "C:\Program Files\Microsoft SDKs\Windows\v7.1\bin\Setenv.cmd" /Release /x86
        call "C:\Program Files\Microsoft SDKs\Windows\v7.1\bin\Setenv.cmd" /Release /x64
```

这里我下载的是 `vs2013 express for Windows Desktop` 版本。安装完成后会提示重启系统，再次安装相应模块即可。

参考资料

- Browsersync / documentation: windows-users
- 用C++为nodejs 写组件，提高node处理效率

Meteor

Meteor是一套用JavaScript开发web和移动端app的工具。再移动平台趋势越来越明显的今天，一套开发环境，多平台支持已经不可缺少了。

参考资料

- Meteor: 官方网站
- Meteor Documentation
- Discover Meteor中文版
- IBM Developer介绍

搭建Meteor开发环境

写这篇文章之前，我按照官方的Toturial走了一遍，发现Meteor的CI(集成工具)真实方便，简单的几个命名之后，一个项目就出来了，好比web开发中的yeoman.

安装

官方暂时也是建议在linux和osx下使用，windows用户最好用虚拟器跑linux测试。安装异常简单：

```
curl https://install.meteor.com/ | sh
```

创建新app

如果需要创建一个名为`simple-todos`的项目，在终端中输入`meteor create simple-todos`即可。

这个命令会创建一个新的叫做`simple-todos`的目录，并且内置了一些文件:

```
simple-todos.js       # a JavaScript file loaded on both client and server
simple-todos.html     # an HTML file that defines view templates
simple-todos.css      # a CSS file to define your app's styles
.meteor               # internal Meteor files
```

运行这个项目也很简单，进入`simple-todos`目录，运行`meteor`即可。Meteor支持一个叫做`hot code push`的技术，就是当你修改代码的时候，页面会自动更新,其实就是实现了f5刷新的功能。。。

关于Template

Meteor使用的模板引擎是`Spacebars`,一个类似`Handlebars`的自己开发的模板引擎。学习资料参见:
Spacebars

Attaching events to templates

通过`Template.templateName.events(...)`这样的形式

Collections

Collections是Meteor的存储方式，创建一个数据库的方式也很简单,`MyCollection = new Mongo.Collection("my-collection");`，在命令行中输入`meteor mongo`即可进入mongodb的shell界面。也可以通过`Robomongo`这样的MongoDB GUI管理界面链接`localhost:3001`进行查看与修改。

Insert

创建好的数据库赋值给了一个全局对象，通过这个对象的`insert()`方法，我们可以向数据库插入数据。

Remove

创建好的数据库赋值给了一个全局对象，通过这个对象的`remove()`方法，我们可以向数据库插入数据。

Update

创建好的数据库赋值给了一个全局对象，通过这个对象的`update()`方法，我们可以向数据库插入数据。

Find

创建好的数据库赋值给了一个全局对象，通过这个对象的`find()`方法，我们可以向数据库查询数据。

部署

在Meteor上注册过账号的，可以使用`meteor deploy my_app_name.meteor.com`命令进行上传，之后就可以通过`my_app_name.meteor.com`进行访问了。

认识Meteor文件结构

上一节中，搭建好了Meteor的开发环境，就可以新建一个项目跑起来。但是新建的项目，没有文件结构层次，显得比较杂乱。

由于Meteor比较灵活，它实际上是自动的帮我们加载了 `script` 和 `link` 标签。

规范的组织方式

- `.meteor` Meteor 存储它内部代码的地方，尝试更改里面的内容并不是什么好主意。但是其中的packages文件和release文件是你安装的所有智能代码包和你使用的Meteor版本信息，值得关注。
- `/client` 此目录下所有的文件都是在客户端使用，一般用于存放HTML, CSS, UI相关的JavaScript代码。
- `/server` 此目录下所有文件都用于服务器端，永远不会用于客户端。一般用来存放敏感的逻辑和数据文件。
- `/lib` 此目录存放的是公共模块，供客户端和服务器端使用。
- `/public` 此目录下文件也是为客户端服务，此目录用于存放比如图片素材，在HTML中直接相对于根目录就能获取到文件。
- `/private` 此目录下文件只为服务器端服务，并且通过 `Assets API` 获取。
- `/client/compatibility` 此目录存放的是兼容性的JavaScript库文件, 这些文件依赖那些全局的变量，在这个目录下的文件会优先于其他客户端脚本文件执行。
- `/packages` 保存自己开发的代码包
- `/test` 此目录下文件永远不会执行，只是为了本地测试

没有按照规范组织的情况

如果JavaScript文件没有放到规范的目录下，都会被加载到客户端和服务器端。Meteor提供了两个方法区分客户端和服务器端:

```
Meteor.isClient() // 在客户端执行
Meteor.isServer() // 在服务器端执行
```

CSS和HTML的文件如果没有按照规范放置的话，只会被加载到客户端，不会加载到服务器端。下面看一个官方的例子：

```
lib/                        # common code like collections and utilities
lib/methods.js              # Meteor.methods definitions
lib/constants.js            # constants used in the rest of the code

client/compatibility        # legacy libraries that expect to be global
client/lib/                 # code for the client to be loaded first
client/lib/helpers.js       # useful helpers for your client code
client/body.html            # content that goes in the <body> of your HTML
client/head.html            # content for <head> of your HTML: <meta> tags, etc
client/style.css            # some CSS code
```

```
client/<feature>.html     # HTML templates related to a certain feature
client/<feature>.js       # JavaScript code related to a certain feature

server/lib/permissions.js # sensitive permissions code used by your server
server/publications.js    # Meteor.publish definitions

public/favicon.ico        # app icon

settings.json             # configuration data to be passed to meteor --settings
mobile-config.js          # define icons and metadata for Android/iOS
```

文件加载顺序

对文件加载顺序最好的执行方法就是尽量避免明确的加载顺序。使用`Meteor.startup`或者在`Packages`中明确加载顺序。

下面是一些关于文件加载顺序的规则：

- HTML模板文件始终第一个加载，在所有文件,即使以`main.`开头的文件
- 以`main.`开头的文件其次加载
- `lib`目录中的文件接着被加载
- 深层次的目录下文件被加载
- 其余文件按照字母顺序依次加载

官方也有一个例子:

```
nav.html
main.html
client/lib/methods.js
client/lib/styles.js
lib/feature/styles.js
lib/collections.js
client/feature-y.js
feature-x.js
client/main.js
```

组织你的项目

比如我们的项目又两个类型的对象`apples`和`oranges`.组织方式通常可以这样：

Root-Level Folders

可以将模块或者对象以顶级目录的形式进行管理:

```
apples/lib/               # code for apple-related features
apples/client/
apples/server/

oranges/lib/              # code for orange-related features
```

```
oranges/client/
oranges/server/
```

Folders inside client/ and server/

也可以分别在不同目录下进行管理:

```
lib/apples/               # common code for apples
lib/oranges/              # and oranges

client/apples/            # client code for apples
client/oranges/           # and oranges

server/apples/            # server code for apples
server/oranges/           # and oranges
```

Packages

最好的管理方式还是形成`packages`，方便独立，模块和重用。

```
packages/apples/package.js     # files, dependencies, exports for apple feature
packages/apples/<anything>.js  # file loading is controlled by package.js

packages/oranges/package.js    # files, dependencies, exports for orange feature
packages/oranges/<anything>.js # file loading is controlled by package.js
```

然后使用publish上传，通过`meteor add <packages>`安装。

参考资料

- Meteor Documentation: File Structure

附录

node相关库

CMS

- Wintersmith 基于Node.js的、灵活的、 简约的、 跨平台静态网站引擎"。当然，"简约"是一个关键词 ！Wintersmith 并不适合于初学者，它比较适合中级或高级开发人员，这样运行起来才不费吹灰之力。Wintersmith 的解决方案并不是体积最小的。它用 CoffeeScript作为开发语言，并内置了 Jade（模板标记语言） 和 Markdown系统。
- Assemble 一个功能强大的工具，将Grunt和Yeoman融入其工作流。像稍后将提到的Punch和DocPad，Assemble 尝试让设计师和程序员协同工作。例如，Assemble预先封装了很多广泛的对初学者友好的模板系统。
- Metalsmith "极简、可插拔的静态网站生成器"。第一次的广告去解释："Metalsmith的所有逻辑都是由插件处理的。你只需要将它们链接在一直。这使得的Metalsmith可以成为这里功能最丰富的系统。
- Hexo Hexo 是目前为止在 GitHub 上最受欢迎的。Hexo的文件简单、美观；包括每一页底部的评论部分。此外，在 GitHub 上你可以找到大量的极小主题，其中大部分是基于或与Ghost 兼容的。
- Punch 它很容易设置，但它给只能呈现出一个近乎空白的页面，当你开始用它工作时。一个基本的启动主题可用，但剩下的就看你的了。
- DocPad 即使你粗略的浏览一下 DocPad 的网站，就能感受到 DocPad 是一个雄心勃勃的项目。你不必看得太深入就能了解到，DocPad与Jekyll的设计理解很相似。DocPad简介页面的底部可以找到显示每个系统必须提供的比较表。DocPad 认为它们是一个功能完整的 CMS 的必要条件。

MongoDB相关库

GUI管理

- Robomongo: Shell-centric cross-platform MongoDB management tool, full platform.

图书资料

教程

- Node.js 包教不包会: 实用性强，篇幅也短。
- 从零开始nodejs系列文章
- 一个月时间整理《深入浅出Node.js》

资源

- awesome-nodejs: A curated list of delightful Node.js packages and resources.

未整理的资料

- How I build Node.js Applications

常用工具

版本控制

- nvm: Simple bash script to manage multiple active node.js versions
- window nvm

命令监控

- forever
- hotnode
- pm2
- nodemon: Monitor for any changes in your node.js application and automatically restart the server - perfect for development